WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, March 10, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L10	l6 and ((plurality or multiple) near2 partitions)	12
☐	L9	6694361.pn. and infiniband	1
☐	L8	6694361.pn.	1
☐	L7	L6 same (look$4 adj (table or list))	17
☐	L6	((send or transmit) near2 (queue or buffer or fifo)) same (receive near2 (queue or buffer or fifo))	3186
☐	L5	(inter-partition near3 (transfer$4 or communication or transmit$4 or send$4))	12
☐	L4	l2 and L3	1
☐	L3	(establish$4 near2 (receive adj (queue or buffer or fifo)))	29
☐	L2	(establish$4 near2 ((send or transmit) adj (queue or buffer or fifo)))	10
☐	L1	(reus$4 near3 (message adj block))	2

END OF SEARCH HISTORY

L5: Entry 6 of 12 File: USPT Nov 6, 2001

DOCUMENT-IDENTIFIER: US 6314501 B1
TITLE: Computer system and method for operating multiple operating systems in different partitions of the computer system and for allowing the different partitions to communicate with one another through shared memory

Brief Summary Text (15):
In one embodiment, the program code that enables inter-partition communication (by managing the shared memory window resources) implements a process by which a sending partition generates an inter-processor interrupt on a receiving partition to signal the receiving partition that information is being transferred to it through the shared memory window. According to this embodiment, the shared memory window comprises a set of input queues associated with each partition, each input queue of the set associated with a given partition corresponding to another partition and storing entries representing communications from that other partition. In order for one partition (a sending partition) to communicate with another partition (a receiving partition), the program code on the sending partition (i) causes an entry to be created in the input queue of the receiving partition that corresponds to the sending partition; and then (ii) causes an inter-processor interrupt to be generated on the receiving partition to signal the receiving partition that the entry has been created in that input queue.

Detailed Description Text (8):
III. Methods for Managing the Global Shared Memory (Inter-Partition Communications)

Detailed Description Text (9):
A. Polling For Inter-Partition Communications

Detailed Description Text (36):
Communications between partitions through shared memory can be managed by any of a variety of methods. In one embodiment, inter-partition communications through shared memory are managed in accordance with an interrupt-driven technique. In another embodiment, a polling technique is used to manage the shared memory communications.

Detailed Description Text (171):
III. Methods for Managing the Global Shared Memory (Inter-Partition Communications)

Detailed Description Text (175):
A. Polling For Inter-Partition Communications

Detailed Description Text (330):
In another exemplary application of the computer system and its global shared memory management, sharing of memory between partitions (i.e., Pods, sub-Pods or operating systems) is achieved while maintaining an appearance of communications by wire. This permits conventional applications programs, conventional application program interfaces (APIs), and conventional communications hardware and software to be used to send data to shared memory. This application is built upon the mechanism described above in Section III.A., in which inter-partition communications are

managed in accordance with a polling technique.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L4: Entry 1 of 1 File: USPT Sep 8, 1998

DOCUMENT-IDENTIFIER: US 5805816 A
TITLE: Network packet switch using shared memory for repeating and bridging packets at media rate

Detailed Description Text (130):
Since the dynamic random access memory 478 is shared between the local area network controllers 452 and 472 via the DMA circuit 480, and the microprocessor 460, the data, address and control buses of memory 478 must be multiplexed to implement this sharing. Likewise, the microprocessor must be able to write data to the local area network controllers at the outset to inform these controllers of the locations of the transmit and receive FIFO buffers which are established in memory 478. The microprocessor also stores the bridge forwarding tables in DRAM 478.

CLAIMS:

4. A packet switching apparatus, comprising:

a shared, multi-port, random access memory having at least first and second ports;

a plurality of media access units for coupling to local area network segments;

a plurality of Local Area Network controller integrated circuits (hereafter LCC) coupled to said media access units and coupled to said first port of said memory, each said LCC for receiving packets from machines coupled to the local area network segments coupled to said media access units and storing them in said memory in a receive buffer, and each said LCC for transmitting via the media access unit coupled to the LCC and the local area network segment coupled to that media access unit any packets having pointers thereto found in a transmit buffer in said main memory assigned to said LCC;

means for establishing in said memory said receive buffer which is large enough to store packets received by all LCC's, and for establishing a transmit buffer for each said LCC and a descriptor ring-having a receive portion and a transmit portion for each said LCC, said receive portion for each LCC for storing status bits indicating when packet reception is complete by that LCC and the packet has been stored in said receive buffer and for placing in a queue in said memory a pointer to the location of said packet in said receive buffer; and

means for monitoring said queue, and, when a pointer is found therein, for accessing the packet pointed to by said pointer and determining if said packet needs to be re-transmitted out on a different local area network segment than the packet arrived on, and, if so, for writing a pointer to said packet into the transmit buffer in said memory assigned to the LCC coupled via a media access unit to the local area network segment upon which said packet needs to be transmitted.

L5: Entry 3 of 12 File: USPT Nov 12, 2002

DOCUMENT-IDENTIFIER: US 6480941 B1
TITLE: Secure partitioning of shared memory based multiprocessor system

Detailed Description Text (10):
FIG. 5 further shows an exemplary memory partitioning. For instance partition P-0 has two regions (561,563), and P-1 and P-2 both have one region mapped, (562) and (564) respectively. It is possible to have certain parts of memory not mapped into any of the partitions (566). Using this scheme, it is also possible to share memory in a non-cache coherent manner between two or more partitions. All that is required is to have overlaps in the memory regions that are defined in the remap table of the various memory controller interfaces. Shared regions can be implemented to provide for inter-partition communication to implement cluster like services.

L5: Entry 5 of 12 File: USPT Nov 27, 2001

DOCUMENT-IDENTIFIER: US 6324588 B1
TITLE: User interface control software for a communication device

Abstract Text (1):
A user interface control software system (125) has software components organized into distinct partitions (210, 220, 230, 240) with a defined interface governing communications between components of different partitions. Preferably, the software components are organized into a core partition (210), a device specific partition (240), a feature enabler partition (220), and a signaling partition (230). The core partition (210) has software components that perform input handling, application interaction management, application control management, and output handling. The device specific partition (240) has software components for interfacing with user interface hardware in a manner specific to a particular type of communication device. The feature enabler partition (220) has software components that implement procedures for interacting with the user to perform a particular task. The signaling partition (230) manages procedures relating to a signaling protocol that requires user interaction. An interface protocol (350) governs inter-partition communications among software components of the various partitions.

Detailed Description Text (2):
The present invention provides for the management of user interface operations through software architected for adaptability across multiple hardware platforms and device configurations. The control software preferably operates on a communication device to manage all user interaction operations. In the preferred embodiment, the control software is architected to facilitate the addition, deletion, and testing of device features and signaling protocols requiring user interaction. The present invention provides for the separation of various components of the control software into distinct partitions with a defined interface governing communications between components of different partitions. Preferably, a set of executable software components are organized into a core partition, a device specific partition, a feature enabler partition, and a signaling partition. The core partition has software components that perform input handling, application interaction management, application control management, and output handling. The device specific partition has software components for interfacing with user interface hardware in a manner specific to a particular type of communication device. The feature enabler partition has software components that implement procedures for interacting with the user to perform a particular task, such as to implement a device feature. The signaling partition manages procedures relating to a signaling protocol that requires user interaction. An interface protocol governs inter-partition communications among software components of the core partition, the device specific partition, the feature enabler partition, and the signaling partition.

Detailed Description Text (11):
The interface protocol 350 governs inter-partition communications among the core partition 210, the feature enabler partition 220, and the signaling partition 230, and between these partitions 210, 220, 230 and the device specific partition 240. The interface protocol 350 permits direct communication between software components of other partitions and the device specific partition. Direct communication is also permitted between software components of the core partition 210 and software

components of the feature enabler partition 220, and between software components of the feature enabler partition 220 and software components of the signaling partition 230. However, in the preferred embodiment, there is no direct interface between software components of the core partition 210 and software components of the signaling partition 230. The interface protocol 350 comprises function calls and messages each having a set of parameters and particular format that specify information content. The interface protocol 350 also specifies rules governing permissible communications among the various partitions. The software components have independent operation with respect to each other except for calls made according to the interface protocol.

L5: Entry 11 of 12 File: USPT Jul 18, 1995

DOCUMENT-IDENTIFIER: US 5434975 A
TITLE: System for interconnecting a synchronous path having semaphores and an asynchronous path having message queuing for interprocess communications

Detailed Description Text (4):
According to the invention, each partition 1-N further includes a hub function 11 that provides asynchronous inter-process communications capability among processes 0-K of that partition 1-N, and provides such capability between processes 0-K of that partition and the processes 0-K of the other partitions 1-N through associated one or more interface nodes 12. The interface nodes 12 are directly interconnected by communication paths 15. Illustratively, each path 15 is a conventional communication mechanism, such as a TCP/IP-protocol link on an Ethernet.RTM. LAN, or a serial point-to-point modem connection. Illustratively, each interface node 12 is implemented as a process in the corresponding partition 1-N. Implementation of interface nodes 12 as processes enables hub functions 11 to treat inter-partition communications, that is, communications to and from interface nodes 12, identically to intra-partition communications between processes 0-K.

L7: Entry 1 of 17 File: USPT Dec 23, 2003

DOCUMENT-IDENTIFIER: US 6667985 B1
TITLE: Communication switch including input bandwidth throttling to reduce output congestion

Detailed Description Text (8):
Each of the RX MACs is associated with a respective input buffer 31 to 34 respectively. These buffers are typically FIFO buffers which may be defined in a respective dedicated section of memory space within a single physical memory device. However they are implemented, each of the input buffers stores in a `receive queue` those packets which have been received by the respective port, pending the forwarding of the packets across the 310 switch to one or more of a multiplicity of transmit queues each of which is held in a respective output buffer (51 to 54) of which there is one for each of the TX MACs 41 to 44. A switch of this general kind customarily has a switching engine 35, a processor (CPU) 36, a look-up table 37, and a statistics function 38. In the schematic illustration of the switch in FIG. 3A the switch is shown with a bus system 39 by means of which data (i e. data packets), status signals and control signals are conveyed from the various functions of the switch to their destinations. It is again emphasised that with the exception of the throttles 21 to 24, switch 30 shown in FIG. 3 is organised in accordance with the state of the art.

L7: Entry 2 of 17 File: USPT Dec 9, 2003

DOCUMENT-IDENTIFIER: US 6661790 B1
TITLE: Packet multicasting in a ring architecture

Detailed Description Text (10):
The device or chip shown in FIG. 4 has a multiplicity of ports 40 which in this example are capable of duplex working, so that each port can both receive packets and send them. Each port is allotted buffer space 41, which may be organised according to known techniques, for example by allotting a dedicated space in SRAM and controlling the loading and unloading of the receive buffer space (RX) and the transmit buffer (TX) by means of respective pointers controlled by a switching ASIC 42. In any convenient manner, packets received on a port 40 are subject to a look-up by means of a look-up table 43 to determine which ports are required for a multicast of the packet. For the sake of conformity with the example of a bit mask given later, the chip is shown as having two ports.

L7: Entry 3 of 17 File: USPT May 27, 2003

DOCUMENT-IDENTIFIER: US 6570848 B1
TITLE: System and method for congestion control in packet-based communication networks

Detailed Description Text (8):
In order to illustrate the process of exerting back pressure, it will be assumed that a packet is to be sent by way of device 11 on switch 1, over the path 3, to port 12 on switch 2 and be forwarded from port 13 on switch 2. The receive path of port 12 receives the packets from device 1 and places them in its static receive buffer 15. A look-up request is sent to the switching engine 19 which by means of the look-up table 21 performs a look-up and determines that the packet is intended for port 13. The link table 22 sets up the necessary link, by way of control of the data bus 18, to enable the sending of the packet in its turn across the switch 2 to the dynamic transmit buffer 16 for port 13.

L7: Entry 6 of 17 File: USPT May 14, 2002

DOCUMENT-IDENTIFIER: US 6389476 B1
TITLE: Networks adapters for multi-speed transmissions

Detailed Description Text (3):
Node 100 includes processor 104, system memory 106, and various I/O adapters 108, 110 interconnected by way of system bus 109, and attaches to one port 101,103 of network 102. Network adapter 110 which processor I/O bus 109 for communication to network 102. Network adapter 110 includes sending adapter 114 which transmits messages from network adapter 110 over port 103 to other network node 111 adapters attached to network 102 at ports 105; receiving adapter 112 which receives messages from the other network node 111 adapters attached by way of ports 105, 101,102 into network adapter 110; and adapter memory 118, including an area of memory dedicated as a send FIFO 115, an area of memory dedicated as a receive FIFO 116, and an area of memory dedicated as a look-up table 117.

Detailed Description Text (4):
In operation, processor 104 sends commands in the form of a 32-bit address word followed by 32-bit data words over I/O bus 109 to control network adapter 110. One of processor 104 commands writes messages directly to send FIFO 115, while another reads messages directly from receive FIFO 116, and yet another writes words to look-up table 117. The address word defines the operation to take place: write to send FIFO 115, read from receive FIFO 116, or write to look-up table 117. Messages for transmission to other nodes are sent from processor 104 over I/O bus 109 to send FIFO 115. After the message is in send FIFO 115, sending adapter 114 controls the reading of the message from send FIFO 115 and transmits it over network 102 at the selected speed. The two preferred embodiments described hereafter in greater detail illustrate the operation of sending adapter 114 for selecting one of four different speeds for the transmission of each message to network 102 over port 103. The speed selection is made individually for each message transmitted.

First Hit Fwd Refs

Generate Collection Print

L10: Entry 1 of 12 File: USPT Oct 7, 2003

DOCUMENT-IDENTIFIER: US 6631421 B1
** See image for Certificate of Correction **
TITLE: Recursive partitioning of networks

Brief Summary Text (12):
In accordance with methods consistent with the present invention, a method is provided that provides a network that has nodes and a plurality of the nodes are partner nodes having a partnership relationship. Furthermore, the method partitions the network into a plurality of subnetworks such that the partnership relationship between the partner nodes remains intact.

Brief Summary Text (13):
In accordance with methods consistent with the present invention, a method is provided in a distributed system having a network of nodes. This method partitions the network into a plurality of subnetworks, provides each of the subnetworks with a routing table that avoids deadlock, and routes traffic through each of the subnetworks using the provided routing tables such that deadlock is avoided.

Detailed Description Text (11):
FIG. 3 depicts a more detailed diagram of routing card 208, although routing card 214 is similarly configured. Routing card 208 contains a memory 302, a switch 304, an d a processor 306 interconnected by an internal bus 307, which also connects to bus 215. Memory 302 contains routing software 308 that routes traffic through the network using routing table 310 and that partitions the network into subnetworks. Routing table 310 may contain part or all of the information contained in the routing tables described below. Memory 302 also contains network topologies 311 and subnetwork routing tables 313. Network topologies 311 include a basic topology that matches each network in the family of networks ranging from 2 to 16 nodes, and for each basic topology, network topology 311 contains various subnetwork topologies reflecting all possible partitionings of the basic topology. Subnetwork routing tables 313 contain deadlock-free routing tables for each subnetwork topology in network topologies 311. Switch 304 coordinates the sending and receiving of information across the network via ports 216-224 by using a send and receive buffer 312-330 for each port.